

```

1  /*
2  Display.cpp - A simple track GPS to SD card logger. Display module.
3  TinyTrackGPS v0.6
4
5  Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
6  All rights reserved.
7
8  raphael.reyes.carmona@gmail.com
9
10 This file is part of TinyTrackGPS.
11
12 TinyTrackGPS is free software: you can redistribute it and/or modify
13 it under the terms of the GNU General Public License as published by
14 the Free Software Foundation, either version 3 of the License, or
15 (at your option) any later version.
16
17 TinyTrackGPS is distributed in the hope that it will be useful,
18 but WITHOUT ANY WARRANTY; without even the implied warranty of
19 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
20 GNU General Public License for more details.
21
22 You should have received a copy of the GNU General Public License
23 along with TinyTrackGPS. If not, see <https://www.gnu.org/licenses/>.
24 */
25
26 #include "Display.h"
27
28 Display::Display(Display_Type t):_screen(t){
29     if (_screen == SDD1306_128X64){
30         _width = 16;
31         _height = 8;
32         _offset = 0;
33     } else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
34         _width = 16;
35         _height = 2;
36         _offset = 0;
37     }
38 }
39
40 void Display::start(){
41     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
42         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
43             // DEFINICION DE CARACTERES PERSONALIZADOS
44             byte alt[8] = {
45                 0b00000100,
46                 0b00001110,
47                 0b00011111,
48                 0b00000100,
49                 0b00000100,
50                 0b00000100,
51                 0b00000100,
52                 0b00000100,
53             };
54
55             byte ant[8] = {
56                 0b00001110,
57                 0b00010001,
58                 0b00010101,
59                 0b00010001,
60                 0b00000100,

```

```

61         0b00000100,
62         0b00001110,
63         0b00000000,
64     };
65
66     byte sd[8] = {
67         0b00001110,
68         0b00010001,
69         0b00011111,
70         0b00000000,
71         0b00000000,
72         0b00010111,
73         0b00010101,
74         0b00011101,
75     };
76
77     byte hourglass_0[8] = {
78         0b00011111,
79         0b00001110,
80         0b00001110,
81         0b00000100,
82         0b00000100,
83         0b00001010,
84         0b00001010,
85         0b00011111,
86     };
87
88     byte hourglass_1[8] = {
89         0b00011111,
90         0b00001010,
91         0b00001110,
92         0b00000100,
93         0b00000100,
94         0b00001010,
95         0b00001010,
96         0b00011111,
97     };
98
99     byte hourglass_2[8] = {
100         0b00011111,
101         0b00001010,
102         0b00001110,
103         0b00000100,
104         0b00000100,
105         0b00001010,
106         0b00001110,
107         0b00011111,
108     };
109
110     byte hourglass_3[8] = {
111         0b00011111,
112         0b00001010,
113         0b00001010,
114         0b00000100,
115         0b00000100,
116         0b00001010,
117         0b00001110,
118         0b00011111,
119     };
120

```

```

121     byte hourglass_4[8] = {
122         0b00011111,
123         0b00001010,
124         0b00001010,
125         0b00000100,
126         0b00000100,
127         0b00001110,
128         0b00001110,
129         0b00011111,
130     };
131     #endif
132     #if defined(DISPLAY_TYPE_LCD_16X2)
133     lcd = new LiquidCrystal(RS, ENABLE, D0, D1, D2, D3);
134     lcd->begin(_width, _height);
135     #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
136     lcd = new LiquidCrystal_I2C(I2C, _width, _height);
137     lcd->init();
138     lcd->backlight();
139     #endif
140
141     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
142     lcd->createChar(0, hourglass_0);
143     lcd->createChar(1, hourglass_1);
144     lcd->createChar(2, hourglass_2);
145     lcd->createChar(3, hourglass_3);
146     lcd->createChar(4, hourglass_4);
147     lcd->createChar(5, alt);
148     lcd->createChar(6, ant);
149     lcd->createChar(7, sd);
150     #endif
151 }
152
153 if (_screen == SDD1306_128X64) {
154     #if defined(DISPLAY_TYPE_SDD1306_128X64)
155     u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
156     u8x8_SSD1306->begin();
157     u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
158     #endif
159 }
160 }
161
162 void Display::clr(){
163     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
164         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
165         lcd->clear();
166         #endif
167     }
168     else if (_screen == SDD1306_128X64) {
169         #if defined(DISPLAY_TYPE_SDD1306_128X64)
170         u8x8_SSD1306->clear();
171         #endif
172     }
173 }
174
175 void Display::print(int vertical, int horizontal, const char text[]){
176     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
177         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
178         lcd->setCursor(vertical, horizontal);
179         lcd->print(text);

```

```

180         #endif
181     }
182     else if (_screen == SDD1306_128X64) {
183         #if defined(DISPLAY_TYPE_SDD1306_128X64)
184             //u8x8_SSD1306->setCursor(vertical, (horizontal*2));
185             //u8x8_SSD1306->print(text);
186             u8x8_SSD1306->setCursor(vertical, (horizontal*2));
187             this->print(text);
188             //u8x8_SSD1306->display();
189             #endif
190     }
191 }
192
193 void Display::print(int line, const char text[]){
194     byte pos = _width -(strlen(text));
195     pos = (pos >> 1);
196     this->print((int)pos, line, text);
197 }
198
199 void Display::print(const char text[]){
200     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
201         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
202             lcd->print(text);
203             #endif
204     }
205     else if (_screen == SDD1306_128X64) {
206         #if defined(DISPLAY_TYPE_SDD1306_128X64)
207             u8x8_SSD1306->print(text);
208             u8x8_SSD1306->flush();
209             #endif
210     }
211 }
212
213 void Display::print(const char text1[], const char text2[]){
214     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
215         this->print(0, text1);
216         this->print(1, text2);
217     }
218     else if (_screen == SDD1306_128X64) {
219         this->print(1, text1);
220         this->print(2, text2);
221     }
222 }
223
224 void Display::print(const char text1[], const char text2[], const char text3[]){
225
226 }
227
228 void Display::print(const char text1[], const char text2[], const char text3[],
229 const char text4[]){
230 }
231
232 void Display::wait_anin(unsigned int t){
233     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
234         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
235             lcd->setCursor(15,1);
236             lcd->write((byte)t%5);
237             #endif
238     }

```

```

239     else if (_screen == SDD1306_128X64) {
240         #if defined(DISPLAY_TYPE_SDD1306_128X64)
241             //char p = 0x2c;
242             //u8x8_SSD1306->drawString((t%16),6,"-");
243
244             uint8_t hourglass_UP[5][8] = { 0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
245                                             0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
246                                             0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
247                                             0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
248                                             0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
249                                             };
250
251             uint8_t hourglass_DOWN[5][8] = {0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
252                                             0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
253                                             0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
254                                             0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
255                                             0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
256                                             };
257             u8x8_SSD1306->drawTile((_width>>1)-1, 6, 1, hourglass_UP[t%5]);
258             u8x8_SSD1306->drawTile((_width>>1)-1, 7, 1, hourglass_DOWN[t%5]);
259         #endif
260     }
261 }
262
263 void Display::print_PChar(byte c) {
264     if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
265         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
266             lcd->write(c);
267         #endif
268     }
269     else if (_screen == SDD1306_128X64) {
270         #if defined(DISPLAY_TYPE_SDD1306_128X64)
271             uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
272                                       0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
273                                       0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
274                                       };
275             uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
276                                         0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
277                                         0x7c,0xfc,0xc0,0xf8,0x7c,0xc0,0xfc,0xf8
278                                         };
279
280             if (c == 5) {
281                 u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
282                 u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
283             }
284             else if (c == 6) {
285                 u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
286                 u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
287             }
288             else if (c == 7) {
289                 u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
290                 u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
291             }
292         #endif
293     }
294 }
295
296 void Display::splash(int time_delay){
297     this->print(NAME, VERSION);
298     delay(time_delay);
299 }

```